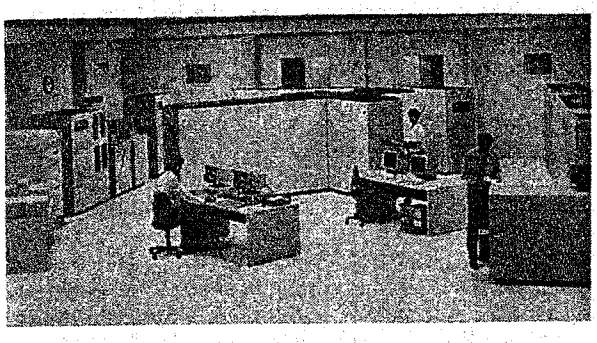

FIG. 1.

The Hardware Subsystem Fig. 1 shows the exterior appearance of a typical very large system's processor complex. Fig. 2 schematically shows the major hardware components of a computer that can be classified into a number of categories:

Transducers Transducers are hardware devices that change information from one physical form to another, linking the computer with its environment. They include keyboards, video screens, printers, plotters and various interactive input devices (*q.v.*) such as the trackball and mouse (*q.v.*).

Storage Devices The same devices store two distinct types of information: 1) data, and 2) instructions (programs). The most elementary unit of stored information is the bit (a 0 or 1 value). For most purposes, a *byte*, which is a group of 8 bits representing 256 possible values, or a character of the system's character-set are treated as an elementary stored unit. Because storages can be very large, certain multiples of a byte are typically used to express storage capacities:

KB = kilobyte	=	1,000 bytes (also 1,024 bytes)
MB = megabyte	=	1,000,000 bytes (also 1,024 KB)
GB = gigabyte	=	1,000,000,000 bytes (also 1,024 MB)
TB = terabyte	=	1,000,000,000,000 bytes (also 1,024 GB)

For economical and technological reasons, storage devices come in many sizes, speeds, and costs. They range from inexpensive, low-capacity, slow devices (e.g. floppy disks - *q.v.*) to larger, more expensive, faster ones (e.g. nonremovable or hard disks (*q.v.*), and tapes), suitable for permanent storage of the information of large commercial, governmental, or educational enterprises.

From a use viewpoint, most storage, which can be hundreds of gigabytes in size, is organized into *files* (*q.v.*) that are retained in the system unless explicitly deleted or replaced. This requires nonvolatile media, such as disk or tape. "Nonvolatile" means that stored contents are retained even if electrical power is removed by shut-down or failure.

Storage with faster access (by a factor of 10,000 or more) than disk storage is physically composed of semiconductor chips and is used in the computer's *main storage* or *memory*. Although far faster than file storage, semiconductor storage is far more expensive per byte and is held to smaller capacities (megabytes). Also, semiconductor memory is *volatile*, meaning that its contents are lost when its power is removed. Although this sounds ominous, in fact it is not, since its contents are needed only during actual processing, and any information that must be retained longer is easily copied to nonvolatile file storage. Because of its speed, main storage, and a related, even faster semiconductor type, called the *cache* or *buffer*, has the special "privilege" of being directly accessible by fast central processing units (CPUs).

Central Processing Units (CPUs) The term "CPU" derives from the fact that for a long time each computer system contained only one processor, and this is the case even today for many smaller systems. The CPU is in many ways the heart of the computer system. It does its work directed by *instructions*, most of which operate on data.

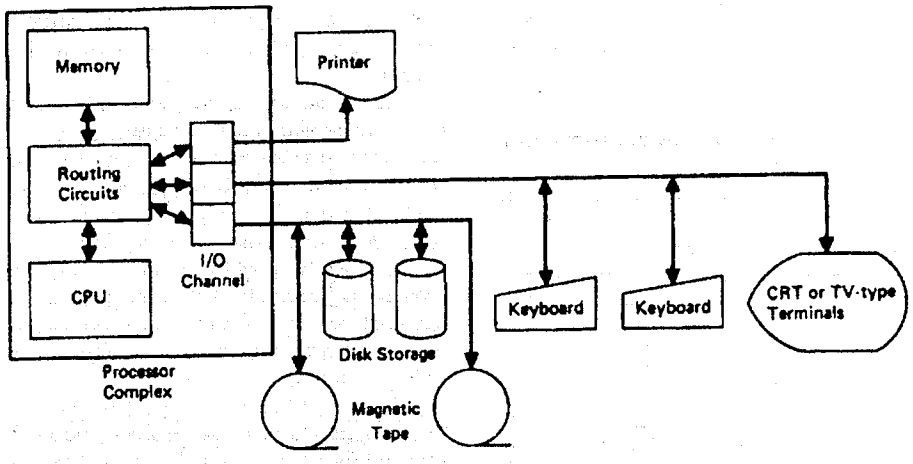

FIG. 2.

The operation performed by each instruction is usually quite primitive. The repertoire of instruction types (e.g. add, subtract, test-for-sign), is called the machine's *instruction set* (*q.v.*) or *instruction repertoire* and typically contains about 200 instructions. But some machines have as few as say 64 instructions in their set, and even this is larger than is theoretically required. The design of a machine's instruction set is an arcane subject with many speed, cost, and even esthetic factors involved (*see* REDUCED INSTRUCTION SET COMPUTER). The instruction-set and associated items constitute the native "machine language" of a computer.

Some CPU circuitry is responsible for execution of the instructions; this includes an adder, augmented by shift and control circuits that together implement the system's arithmetic and processing operations. The ability to execute all instructions, while necessary, is *not* sufficient. Also needed is a control mechanism to *sequence* from one instruction to the next in a program of instructions stored in the computer's main storage. Programs (i.e. specific instruction sequences) express the required function of the program (payroll, inventory, etc.).

Although not an exact analogy, the instruction-set is akin to a set of musical notes, as is found on a piano, and a program is like the score for a particular song. In the case of music, the human performer "reads" the next note from the score, then invokes that note from the instrument, then repeats the process for the next note, etc. In the computer case, the CPU contains *program control circuitry* that automatically reads or accesses the stored program's next instruction, and then invokes the computer's execution circuitry to execute the instruction. This cycle is then repeated automatically. During execution, many instructions call for data, which are held in the same main storage as the program. Main storage appears as a long list of memory cells, each with a unique location-number called its *address*. So main storage is addressed for instructions that in turn address it for data.

Also essential for program control are provisions for testing conditions on input or computed data and, depending on the test-outcome, *branching* or jumping in the stored program sequence from strict progression to a new program segment. Using branching, a single segment of stored program may be used repetitively thousands of times in the same program, with certain necessary program-modifiable differences in each such use.

All CPU functions use the fastest (but expensive, and hence small) kind of storage—the CPU registers—usually built from the same circuitry used for processing. The registers are a sort of scratch pad for the CPU to jot down results, especially those used frequently in local parts of the program. Some register contents are later transferred to main storage and even to file storage.

Routing and Control Circuitry Routing circuits include the networks and "buses" that direct the flow of information between various functional parts of the hardware subsystem (*see* BUS). For instance, the I/O channels control the flow of information between the transducers, peripheral storage devices, and main storage. Other routing circuits control communication between main storage and the CPU. The control circuitry generates timing signals in various arrangements that specify at what times which information is moved from place to place in the system.

Another classification scheme divides the hardware subsystem into "internal" and "external" items. The internal ones are the CPU, with its registers, instruction-execution and control units, main storage and cache storage, and I/O channels. This "internal computer," which may contain several CPUs sharing access to common main storage and sets of channels is sometimes called the "processor complex" (see Fig 1). All other hardware devices, such as transducers and file-storage disks and tapes, are referred to as the *peripheral*, or input/output, or *I/O*, subsystem.

General Hardware Organization How do the constituents of these hardware categories work together? The reader is advised to trace the following description through the paths and facilities of Fig. 2. The program (say, as a sequence of keystrokes entered by the programmer) must first be physically translated into electric-signal form, which is done by the terminal keyboard, a transducer. Once in electrical form, the typed information is stored as a file on nonvolatile disk storage for future use. Accordingly, it is moved by an I/O channel to disk storage (briefly passing through main storage in the process), where it is held until it is to be executed.

To be executed, the program must satisfy two main requirements: it must be in machine-language (object-code) form, and it must reside in main storage. As to the first, the typed form of the program (source code) must almost always be translated and then typically stored as an object-code file. Assume for the moment that this has been done, so it remains to move the object-code file from disk storage to memory via an I/O channel and routing circuits.

Once in memory, the object-code program is executable by the CPU. During execution, most storage accessing is to the memory (main store). However, the running program is capable of directing movement of data, including final results between memory and peripheral storages (disk or tape) via the I/O channels. Later, such results can be moved, again via channels to a transducer (display terminal or printer) for human inspection.

Until about 1968, most computers were *mainframes* (*q.v.*) that were expensive and required considerable floor space, electric power, and air conditioning. However, due to rapid advances in the computer technologies, especially semiconductor circuitry, by the late 1960s a genre of small-size machines termed *minicomputers* (*q.v.*) started to be produced that occupied only the space of an office desk, and were inexpensive enough to be used for the solution of a single problem, such as the control of particular industrial process as well as a variety of shared uses.

By about 1973, further rapid advances in technology led to another class of even smaller, cheaper systems based on the *microprocessor*, a complete CPU contained on a single or very few semiconductor chips. This method of manufacture eliminated most hand-wiring and sepa-

rate-component manufacture of the CPU with substantial improvements, not only in cost but also reliability.

This great advance in technology was matched by corresponding advances in semiconductor memory that allowed compact, high-capacity main storage to be offered at a price hundreds of times cheaper per unit of stored information than was available only a decade earlier. By 1990, several million microprocessor-based systems, called *personal computers* (*PCs* - *q.v.*) or *workstations* (*q.v.*) were in common use in homes, offices, and businesses. Each supplied to an individual user computation power and main storage that had been available in shared form only from large, expensive systems a few years earlier. Furthermore, the progress in technology that has given us the microprocessor has also enhanced the cost effectiveness of the larger minicomputer and general-purpose mainframe systems.

Here, only a very brief account has been given of three main classes of computer system: mainframe, minicomputer, and microprocessor-based. These are not precise categories. Thus, many minicomputers are shared by several users and applications in the same way as are mainframe systems. Although each PC/workstation is typically used by only one person, some are being interconnected via *local area networks* (LANs - *q.v.*) that also contain *servers* that use microprocessor-based technology to permit access to large shared disk storage by the many PCs and workstations on the LAN (*see* FILE SERVER).

The Software Subsystem Unlike hardware, software (*q.v.*) is not tangible. Software, although held in a physical medium, say on a disk storage unit, is composed of programs and data arranged in logical, not physical, structures (*see* DATA STRUCTURES).

Software is usually described in two major categories, *application software* and *system software*, with subcategories:

1. Application software:
   a. Programs written by users for a particular purpose, such as payroll, inventory control, design of a product, etc.
   b. "Packaged" programs written and supplied by vendors for end-users, each for a wide but restricted range of purposes. Examples: word processors for creating text documents; spreadsheets (*q.v.*) for financial analysis work; database packages for creating, maintaining, and searching large structured data collections; statistical analysis packages, etc.
   c. Installation libraries, containing programs of types a. and b. and databases that are particular to several users at a site or an enterprise. Increasingly, a central "repository," or "data dictionary," itself an item of complex software, is used to manage access to these objects.
2. System software:
   a. Operating system.
   b. Language processors (compilers).
   c. Utilities.

The following discussion concentrates on system software, which is part of the system itself.

The *operating system* is usually the most complex software in a computer system. Two of its basic purposes are: 1) to supply common functions or "services" for use by other software, and 2) to control the orderly *sharing* of the system's hardware and software resources by several users where such sharing is done. Most of the types of system programs below are part of the operating system:

1. Device drivers (DD)—There is one per device that controls device-specific details of a video screen, a printer, a disk storage, etc. Most other software will, as needed, call on a DD when it interacts with the device. Such software sees the device as a much simpler logical unit than is seen by the DD itself, and in this way most software is shielded from onerous device-sensitive details.
2. Data management programs (DMP)—These keep track of the named storage items such as files, i.e. where each is located and the means to store and access the data efficiently. For instance, when a user's program calls for data, a DMP locates and fetches the data to the requesting program. (In so doing, the DMP may well call on the DD part of the operating system described earlier). For each file, the DMP may maintain information as to who is permitted to use the data, who is currently using it, what is being done with it, whether or not the data should be retained in the system after the job ends, etc. Some parts of DMP software are called *access methods* (*q.v.*).
3. Linkers/loaders—These programs do the final preparation of object code programs prior to initiation of execution. Included may be "binding" or linking of references in one program with another using machine-language naming (addresses).
4. System control program (SCP)—This complex part of operating system software is found in *shared systems*, as is common in most mainframe and minicomputer systems, but not in many personal computers or workstations that serve single users only.
5. Language processors (translators/compilers)—Programs that are executed by the computer must be in machine-language form, which is exceedingly tedious for humans to use when creating programs. Accordingly, modern computer systems support much more convenient *higher-level languages* (HLL) for human use, along with language translators, usually compilers, to deal with them. These translate user-written programs (*source code*) from such HLL source languages as Cobol or Fortran or C (*q.v.*) into machine-language object code, the only form the machine can execute.
6. Utilities—These are programs that perform frequently required tasks, such as sorting (*q.v.*) and